

```
!pip install pyspark
```

```
Requirement already satisfied: pyspark in /usr/local/lib/python3.12/dist-packages (4.0.1)
Requirement already satisfied: py4j==0.10.9.9 in /usr/local/lib/python3.12/dist-packages (from pyspark) (0.10.9.9)
```

```
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName("AI Impact on Student Performance") \
    .getOrCreate()

spark
```

SparkSession - in-memory

SparkContext

[Spark UI](#)

```
Version
      v4.0.1
Master
      local[*]
AppName
      AI Impact on Student Performance
```

```
!find / -name "ai_impact_student_performance_dataset.xlsx" 2>/dev/null
```

```
/ai_impact_student_performance_dataset.xlsx
```

▼ Step 1: Load Dataset

We read the Excel dataset containing student performance and AI usage information.

```
import pandas as pd

# Correct path after uploading in Colab
excel_path = "/ai_impact_student_performance_dataset.xlsx"
df_pandas = pd.read_excel(excel_path)

# Check first 5 rows
df_pandas.head()
```

	student_id	age	gender	grade_level	study_hours_per_day	uses_ai	ai_usage_time_minutes	ai_tools_used	ai_usage_purpo
0	1	20	Female	1st Year	2.5	1	170	NaN	Exam Prepa
1	2	17	Male	12th	3.4	1	123	NaN	No Class
2	3	24	Male	3rd Year	0.8	0	35	Copilot	Doubt Solv
3	4	21	Female	12th	4.4	0	45	ChatGPT+Gemini	No Class
4	5	18	Other	3rd Year	3.5	1	21	ChatGPT+Gemini	Coding

5 rows × 26 columns

```
!mkdir -p data
```

```
# Save CSV in the project folder (portable for submission)
csv_path = "data/ai_impact_student_performance_dataset.csv"
df_pandas.to_csv(csv_path, index=False)
print("Saved to:", csv_path)
```

```
Saved to: data/ai_impact_student_performance_dataset.csv
```

```
from google.colab import files
files.download("data/ai_impact_student_performance_dataset.csv")
```

```
from pyspark.sql import SparkSession

# Start Spark session
spark = SparkSession.builder \
    .appName("AI Impact on Student Performance") \
    .getOrCreate()

# Load the CSV
csv_path = "data/ai_impact_student_performance_dataset.csv"
df_spark = spark.read.csv(csv_path, header=True, inferSchema=True)

# Show first 5 rows
df_spark.show(5)

# Check number of rows and columns
print("Total Rows:", df_spark.count())
print("Total Columns:", len(df_spark.columns))
```

student_id	age	gender	grade_level	study_hours_per_day	uses_ai	ai_usage_time_minutes	ai_tools_used	ai_usage_purpose	ai_dependency_score
1	20	Female	1st Year	2.5	1	170	NULL	Exam Prep	
2	17	Male	12th	3.4	1	123	NULL	Notes	
3	24	Male	3rd Year	0.8	0	35	Copilot	Doubt Solving	
4	21	Female	12th	4.4	0	45	ChatGPT+Gemini	Notes	
5	18	Other	3rd Year	3.5	1	21	ChatGPT+Gemini	Coding	

only showing top 5 rows

Total Rows: 8000

Total Columns: 26

```
df = spark.read.csv(
    csv_path,
    header=True,
    inferSchema=True
)

df.show(5)
```

student_id	age	gender	grade_level	study_hours_per_day	uses_ai	ai_usage_time_minutes	ai_tools_used	ai_usage_purpose	ai_dependency_score
1	20	Female	1st Year	2.5	1	170	NULL	Exam Prep	
2	17	Male	12th	3.4	1	123	NULL	Notes	
3	24	Male	3rd Year	0.8	0	35	Copilot	Doubt Solving	
4	21	Female	12th	4.4	0	45	ChatGPT+Gemini	Notes	
5	18	Other	3rd Year	3.5	1	21	ChatGPT+Gemini	Coding	

only showing top 5 rows

```
print("Total Rows:", df.count())
print("Total Columns:", len(df.columns))
```

Total Rows: 8000

Total Columns: 26

```
df.printSchema()
df.show(5)
```

```
root
|-- student_id: integer (nullable = true)
|-- age: integer (nullable = true)
|-- gender: string (nullable = true)
|-- grade_level: string (nullable = true)
|-- study_hours_per_day: double (nullable = true)
|-- uses_ai: integer (nullable = true)
|-- ai_usage_time_minutes: integer (nullable = true)
|-- ai_tools_used: string (nullable = true)
|-- ai_usage_purpose: string (nullable = true)
|-- ai_dependency_score: integer (nullable = true)
|-- ai_generated_content_percentage: integer (nullable = true)
|-- ai_prompts_per_week: integer (nullable = true)
|-- ai_ethics_score: integer (nullable = true)
|-- last_exam_score: integer (nullable = true)
|-- assignment_scores_avg: double (nullable = true)
|-- attendance_percentage: double (nullable = true)
```

```
-- concept_understanding_score: integer (nullable = true)
-- study_consistency_index: double (nullable = true)
-- improvement_rate: double (nullable = true)
-- sleep_hours: double (nullable = true)
-- social_media_hours: double (nullable = true)
-- tutoring_hours: double (nullable = true)
-- class_participation_score: integer (nullable = true)
-- final_score: double (nullable = true)
-- passed: integer (nullable = true)
-- performance_category: string (nullable = true)

+-----+-----+-----+-----+-----+-----+-----+-----+
|student_id|age|gender|grade_level|study_hours_per_day|uses_ai|ai_usage_time_minutes|ai_tools_used|ai_usage_purpose|ai_depe
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1| 20|Female| 1st Year| 2.5| 1| 170| NULL| Exam Prep|
| 2| 17| Male| 12th| 3.4| 1| 123| NULL| Notes|
| 3| 24| Male| 3rd Year| 0.8| 0| 35| Copilot| Doubt Solving|
| 4| 21|Female| 12th| 4.4| 0| 45|ChatGPT+Gemini| Notes|
| 5| 18| Other| 3rd Year| 3.5| 1| 21|ChatGPT+Gemini| Coding|
+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 5 rows
```

```
df_clean = df.dropna()
```

▼ Step 2: Data Cleaning

We drop rows with missing values to ensure accurate analysis.

```
from pyspark.sql.functions import avg

df_clean.agg(
    avg("final_score").alias("avg_final_score")
).show()
```

```
+-----+
| avg_final_score|
+-----+
|56.93036231884044|
+-----+
```

```
df_clean.groupBy("uses_ai") \
    .agg(avg("final_score").alias("avg_final_score")) \
    .show()
```

```
+-----+
|uses_ai| avg_final_score|
+-----+
| 1|56.88801252847375|
| 0|57.00443227091633|
+-----+
```

```
df_clean.groupBy("performance_category") \
    .agg(avg("ai_dependency_score").alias("avg_ai_dependency")) \
    .show()
```

```
+-----+
|performance_category|avg_ai_dependency|
+-----+
| High|5.573863636363637|
| Low|5.605691056910569|
| Medium|5.437614678899083|
+-----+
```

```
df_clean.groupBy("study_hours_per_day")
```

```
GroupedData[grouper expressions: [study_hours_per_day], value: [student_id: int, age: int ... 24 more fields], type: GroupBy]
```

```
from pyspark.sql.functions import avg

df_clean.groupBy("study_hours_per_day")\
    .agg(avg("final_score").alias("avg_score"))\
```

```
.orderBy("study_hours_per_day")\
.show()
```

study_hours_per_day	avg_score
0.5	55.42619047619049
0.6	55.09722222222223
0.7	55.58804347826089
0.8	58.14324324324324
0.9	55.38148148148148
1.0	55.7959595959595961
1.1	54.62134831460672
1.2	57.8528735632184
1.3	55.34022988505749
1.4	57.57340425531915
1.5	58.62826086956521
1.6	56.50510204081635
1.7	57.99626168224297
1.8	57.79652173913042
1.9	56.6879120879121
2.0	57.73725490196078
2.1	57.8541666666666664
2.2	55.856730769230765
2.3	56.141052631578965
2.4	56.82499999999999

only showing top 20 rows

```
df_clean.groupBy("uses_ai")\
.agg(avg("final_score").alias("avg_final_score"))\
.show()
```

uses_ai	avg_final_score
1	56.88801252847375
0	57.00443227091633

```
df_clean.groupBy("performance_category")\
.agg(avg("ai_dependency_score").alias("avg_ai_dependency"))\
.show()
```

performance_category	avg_ai_dependency
High	5.5738636363636367
Low	5.605691056910569
Medium	5.437614678899083

```
df_clean.groupBy("gender")\
.agg(avg("final_score").alias("avg_final_score"))\
.show()
```

gender	avg_final_score
Female	56.86914569031269
Other	56.00533980582523
Male	57.060772659732585

```
df_clean.select(
    "study_hours_per_day",
    "sleep_hours",
    "social_media_hours",
    "tutoring_hours",
    "final_score"
).toPandas().corr()
```

	study_hours_per_day	sleep_hours	social_media_hours	tutoring_hours	final_score
study_hours_per_day	1.000000	0.021906	0.004877	-0.004978	0.019213
sleep_hours	0.021906	1.000000	0.007303	0.022342	-0.014562
social_media_hours	0.004877	0.007303	1.000000	-0.016728	0.001144
tutoring_hours	-0.004978	0.022342	-0.016728	1.000000	-0.000730
final_score	0.019213	-0.014562	0.001144	-0.000730	1.000000

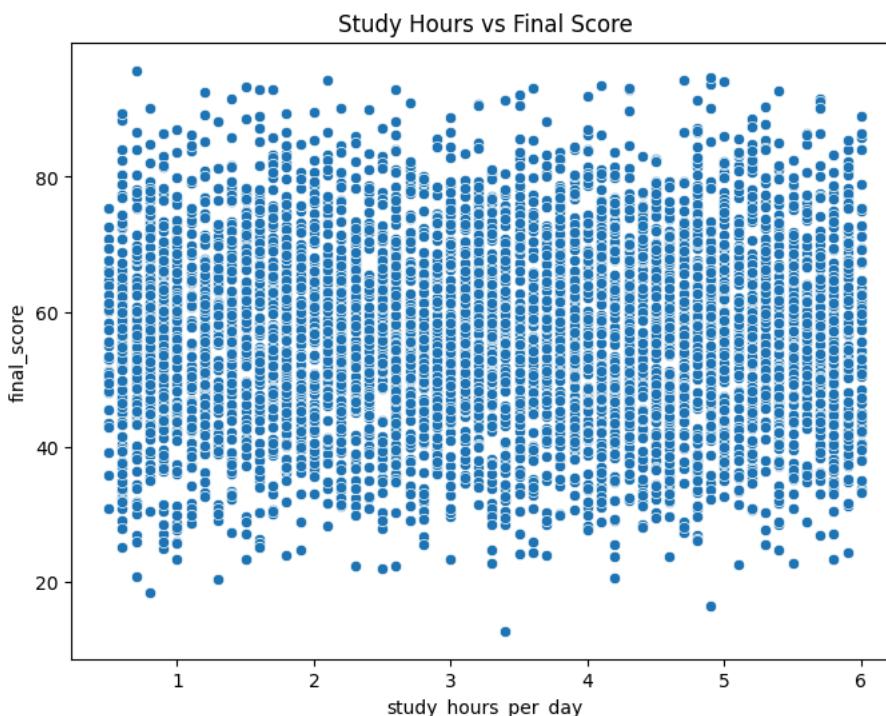
```
df_clean.groupBy("ai_tools_used")\
    .agg(avg("final_score").alias("avg_final_score"))\
    .orderBy("avg_final_score", ascending=False)\n    .show()
```

```
+-----+-----+
| ai_tools_used| avg_final_score|
+-----+-----+
| Gemini|57.29255222524988|
| ChatGPT+Gemini|57.03666092943195|
| Copilot|56.88929577464779|
| ChatGPT|56.79248658318433|
| Claude| 56.6283054003725|
+-----+-----+
```

▼ Step 3: Exploratory Analysis

We explore numeric and categorical features, compute correlations, and visualize distributions.

```
import matplotlib.pyplot as plt\nimport seaborn as sns\n\n# Convert to Pandas\npdf = df_clean.select("study_hours_per_day", "final_score").toPandas()\n\n# Scatter plot\nplt.figure(figsize=(8,6))\nsns.scatterplot(x="study_hours_per_day", y="final_score", data=pdf)\nplt.title("Study Hours vs Final Score")\nplt.show()
```



```
import matplotlib.pyplot as plt\nimport seaborn as sns\n\n# Convert Spark DataFrame to Pandas for plotting\npdf = df_clean.toPandas()
```

```
# 1 Scatter plots: Numeric features vs final_score
numeric_cols = ["study_hours_per_day", "sleep_hours", "social_media_hours", "tutoring_hours"]

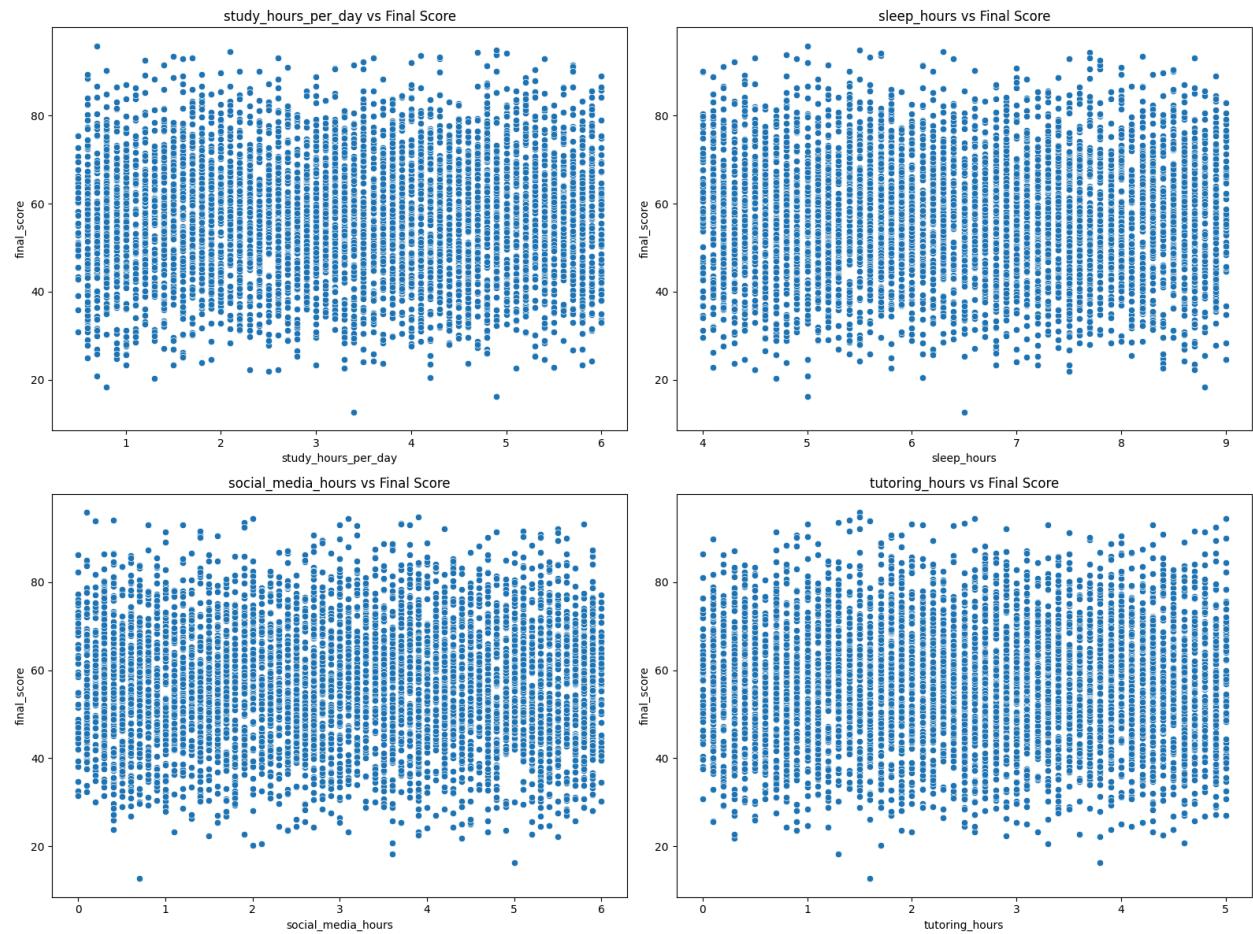
plt.figure(figsize=(16,12))
for i, col in enumerate(numeric_cols, 1):
    plt.subplot(2, 2, i)
    sns.scatterplot(x=col, y="final_score", data=pdf)
    plt.title(f"{col} vs Final Score")
plt.tight_layout()
plt.show()

# 2 Correlation heatmap for numeric columns
plt.figure(figsize=(12,10))
corr = pdf[numeric_cols + ["final_score"]].corr()
sns.heatmap(corr, annot=True, fmt=".2f", cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()

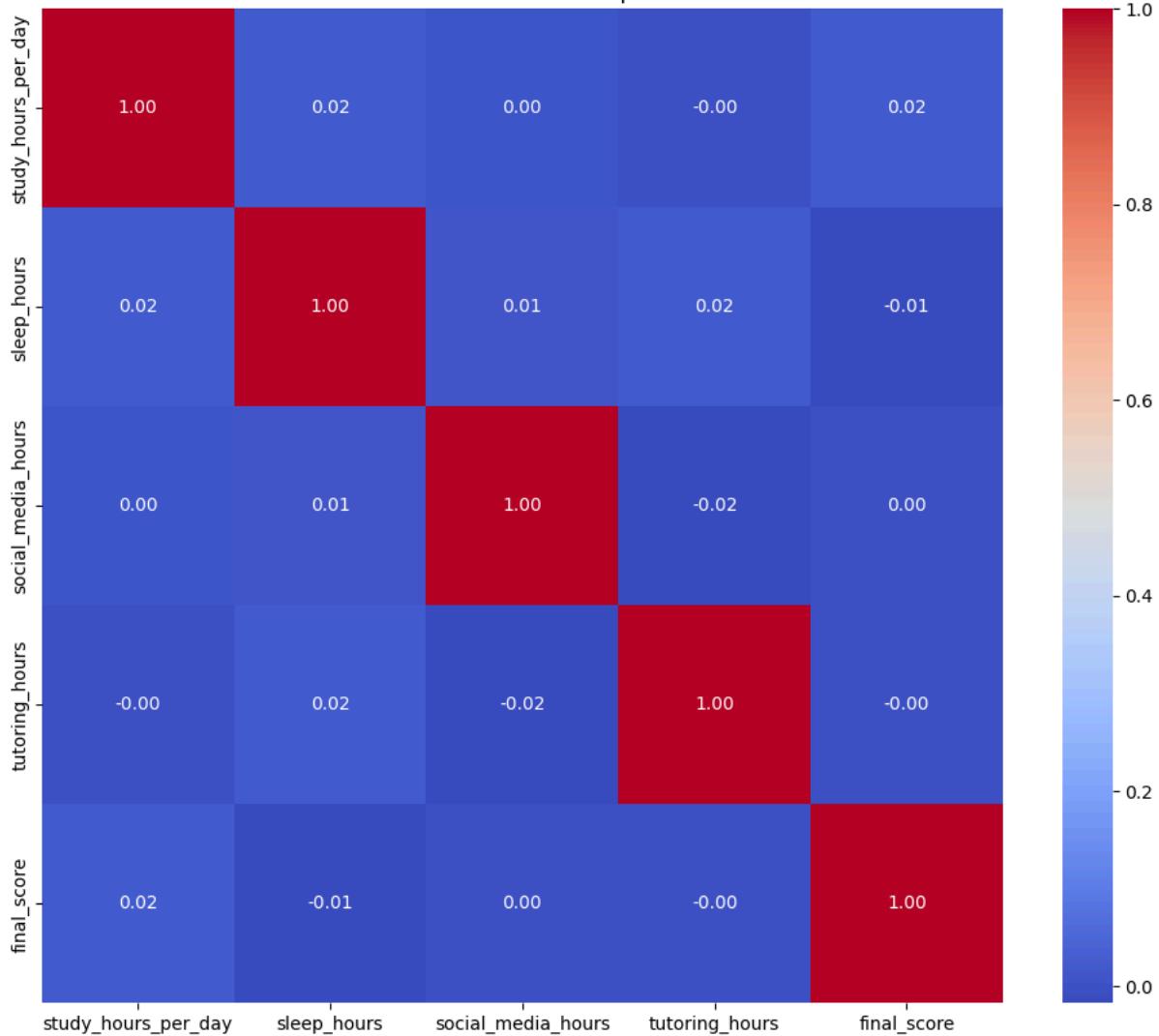
# 3 Bar plots: Categorical features vs avg final_score
categorical_cols = ["uses_ai", "performance_category", "gender", "ai_tools_used"]

for col in categorical_cols:
    plt.figure(figsize=(8,5))
    sns.barplot(x=col, y="final_score", data=pdf, ci=None)
    plt.title(f"Average Final Score by {col}")
    plt.xticks(rotation=45)
    plt.show()

# 4 Boxplots for distribution insights
for col in categorical_cols:
    plt.figure(figsize=(8,5))
    sns.boxplot(x=col, y="final_score", data=pdf)
    plt.title(f"Final Score Distribution by {col}")
    plt.xticks(rotation=45)
    plt.show()
```

Correlation Heatmap

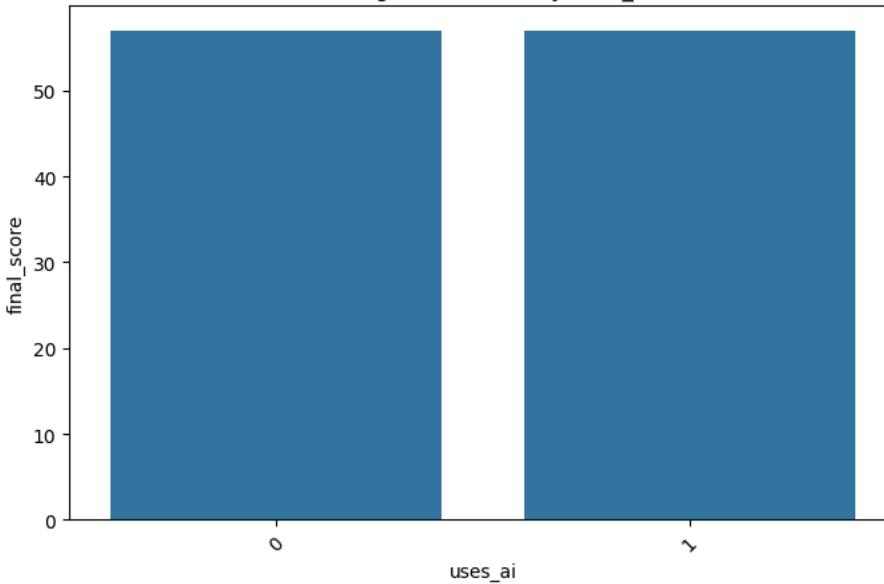


```
/tmp/ipython-input-2848995358.py:36: FutureWarning:
```

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

```
sns.barplot(x=col, y="final_score", data=pdf, ci=None)
```

Average Final Score by uses_ai

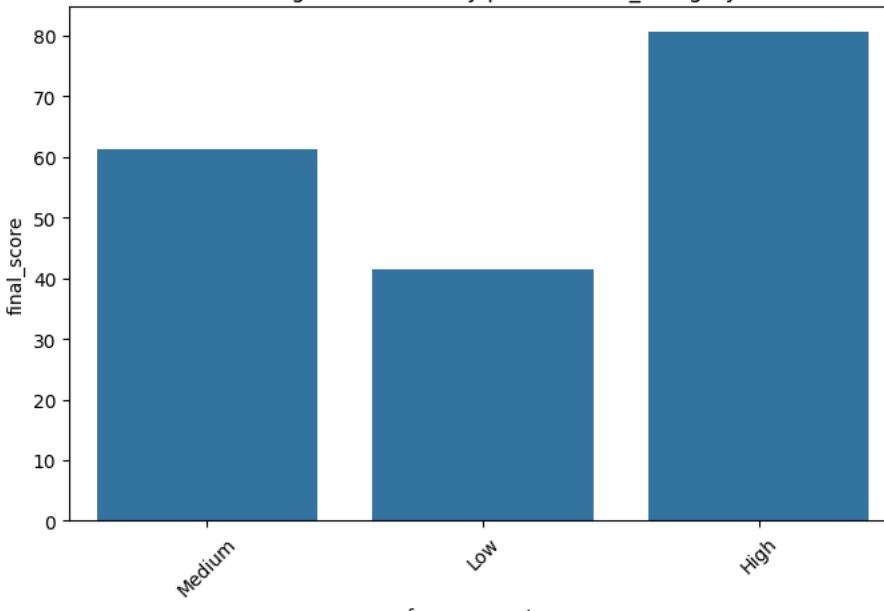


/tmp/ipython-input-2848995358.py:36: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

```
sns.barplot(x=col, y="final_score", data=pdf, ci=None)
```

Average Final Score by performance_category

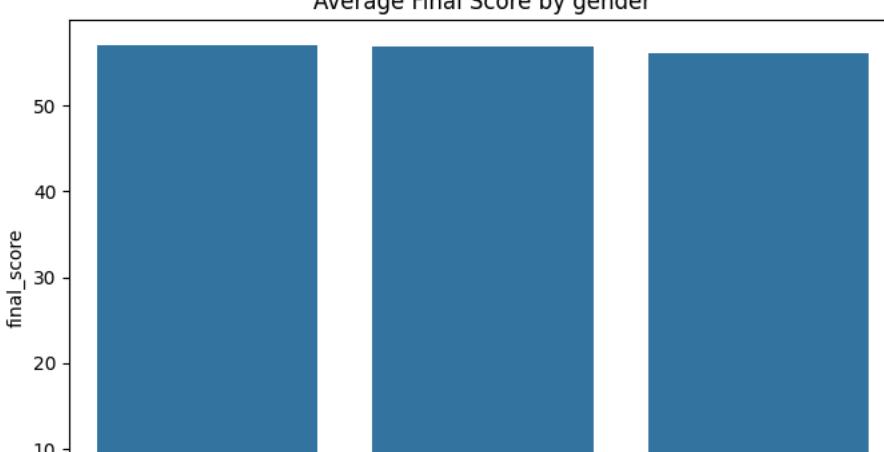


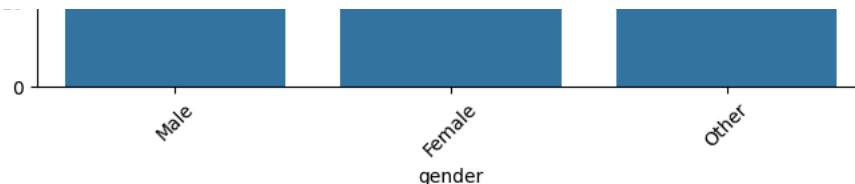
/tmp/ipython-input-2848995358.py:36: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

```
sns.barplot(x=col, y="final_score", data=pdf, ci=None)
```

Average Final Score by gender

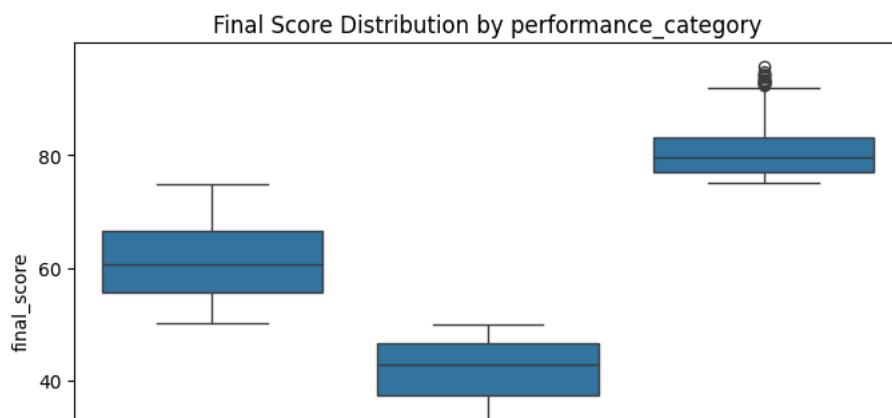
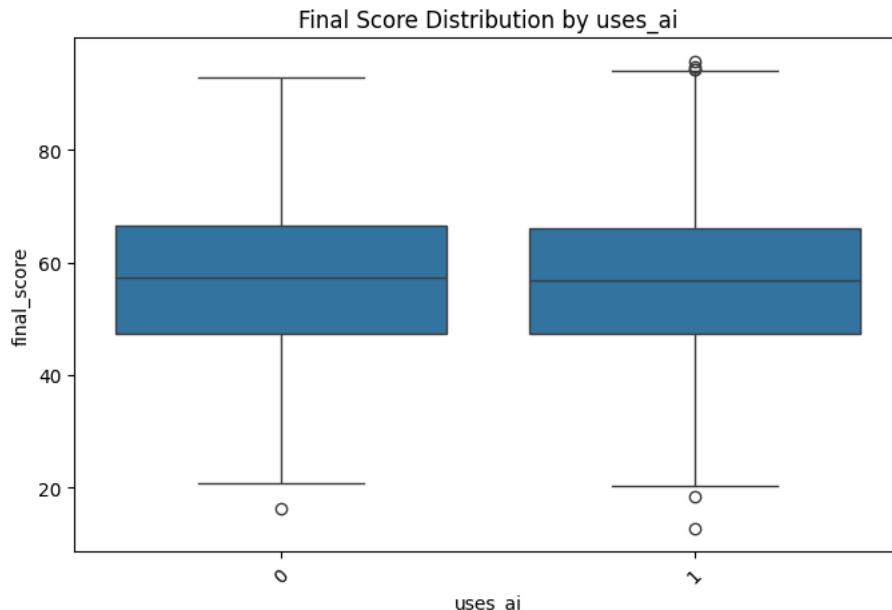
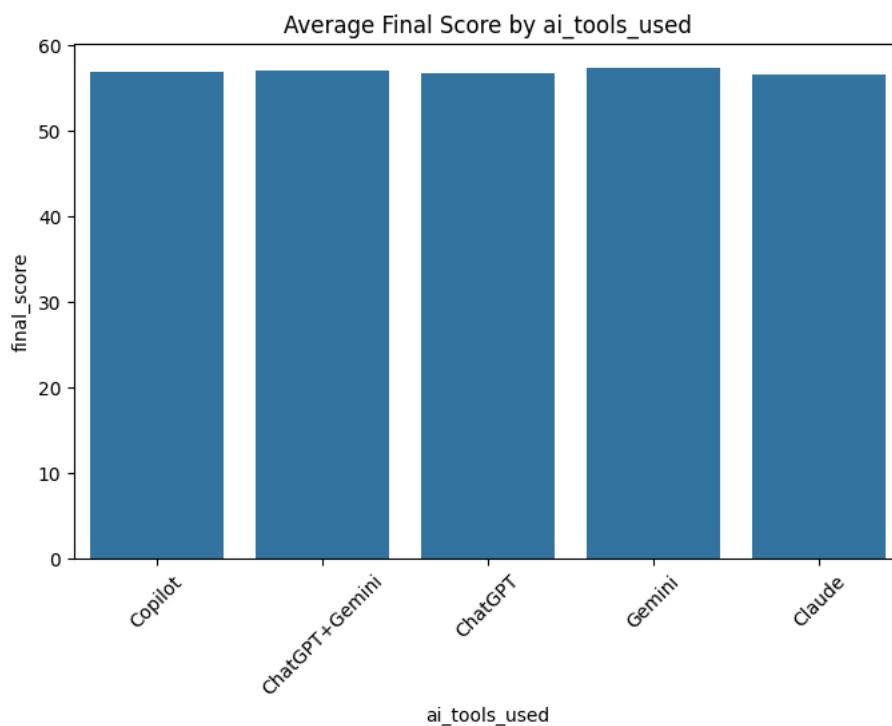


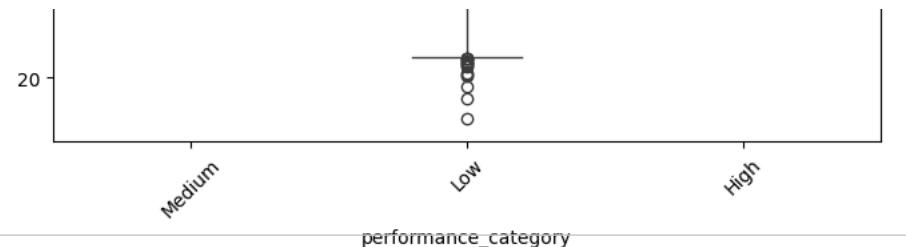


```
/tmp/ipython-input-2848995358.py:36: FutureWarning:
```

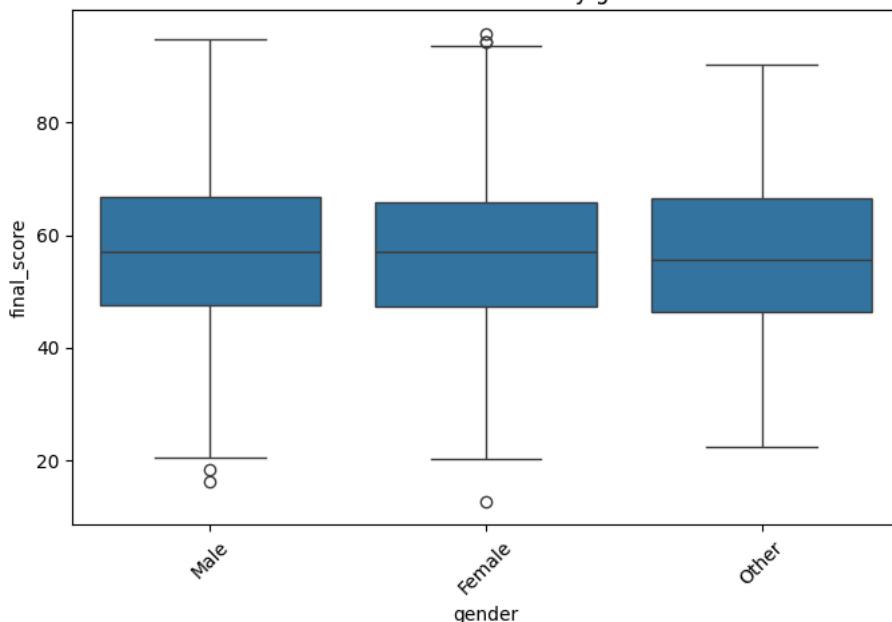
The `ci` parameter is deprecated. Use `errorbar=None` for the same effect.

```
sns.barplot(x=col, y="final_score", data=pdf, ci=None)
```

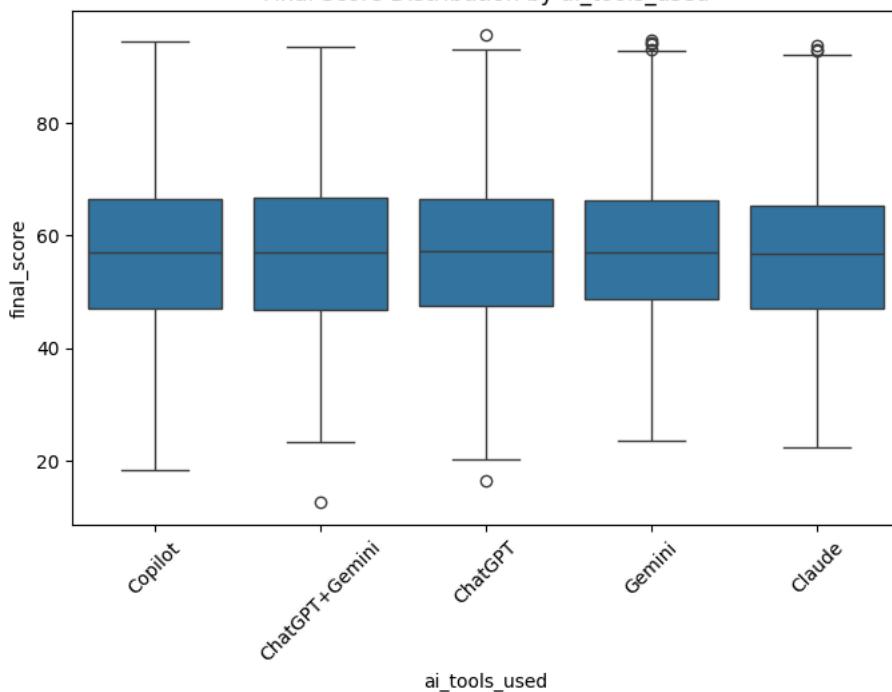




Final Score Distribution by gender



Final Score Distribution by ai_tools_used



✓ Step 4: Correlation Analysis

We calculate correlations of numeric features with the final score.

```
# Correlation Analysis

# Select numeric columns only
numeric_cols = ["study_hours_per_day", "sleep_hours", "social_media_hours",
                 "tutoring_hours", "ai_dependency_score", "ai_prompts_per_week",
                 "ai_ethics_score", "last_exam_score", "assignment_scores_avg",
                 "attendance_percentage", "concept_understanding_score",
                 "study_consistency_index", "improvement_rate", "final_score"]

# Convert Spark DF to Pandas for correlation
pdf = df_clean.select(numeric_cols).toPandas()

# Compute correlation with final_score
corr_with_final = pdf.corr()["final_score"].sort_values(ascending=False)
print("Correlation of numeric features with final_score:\n")
print(corr_with_final)
```

Correlation of numeric features with final_score:

```
final_score          1.000000
last_exam_score      0.682541
assignment_scores_avg 0.455588
concept_understanding_score 0.426480
study_hours_per_day   0.019213
ai_ethics_score       0.006225
social_media_hours    0.001144
tutoring_hours        -0.000730
improvement_rate      -0.002192
attendance_percentage -0.003586
ai_dependency_score   -0.004225
ai_prompts_per_week   -0.006044
study_consistency_index -0.011094
sleep_hours            -0.014562
Name: final_score, dtype: float64
```

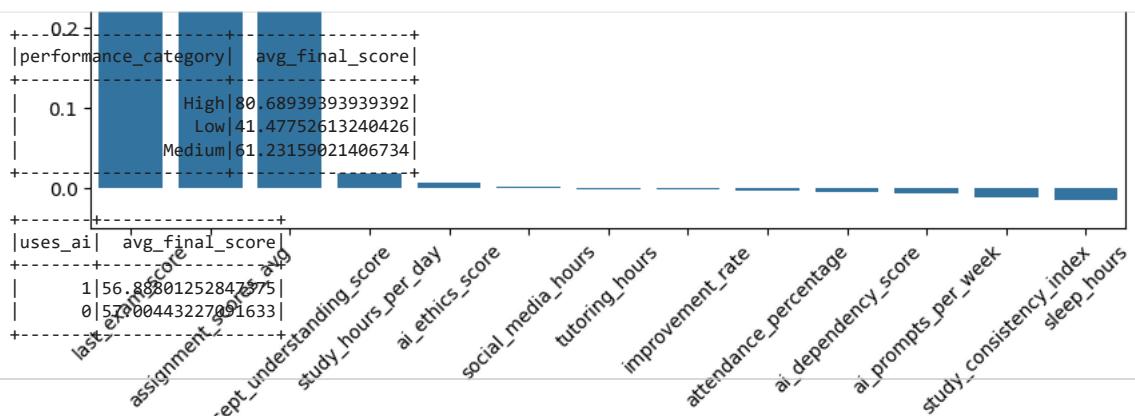
```
# Visualize top correlated features
top_features = corr_with_final.drop("final_score").sort_values(ascending=False)

plt.figure(figsize=(10,6))
sns.barplot(x=top_features.index, y=top_features.values)
plt.title("Correlation of Features with Final Score")
plt.ylabel("Correlation")
plt.xticks(rotation=45)
plt.show()
```

Correlation of Features with Final Score

```
# Average final_score by performance_category
df_clean.groupBy("performance_category")\
    .agg(avg("final_score").alias("avg_final_score"))\
    .show()
```

```
# By AI usage
df_clean.groupBy("uses_ai")\
    .agg(avg("final_score").alias("avg_final_score"))\
    .show()
```



▼ Step 5: Insights & Summary

None

We summarize key findings from the analysis and highlight important trends.

```
# Insights & Summary

print("===== BIG DATA ANALYSIS: AI Impact on Student Performance =====\n")

# Total students analyzed
print("Total students analyzed:", df_clean.count())
print("Total features analyzed:", len(df_clean.columns), "\n")

# Overall average final score
overall_avg = df_clean.agg(avg("final_score").alias("avg_final_score")).collect()[0][0]
print(f"Overall Average Final Score: {overall_avg:.2f}")

# Average final score by AI usage
ai_avg = df_clean.groupBy("uses_ai").agg(avg("final_score").alias("avg_final_score")).collect()
print("\nAverage Final Score by AI Usage:")
for row in ai_avg:
```